Colored props for large scale graphical reasoning

Titouan Carette, Simon Perdrix

Colored props for large scale graphical reasoning

Titouan Carette
CNRS, LORIA, Inria Mocqua, Université de Lorraine, F 54000 Nancy, France
titouan.carette@loria.fr

Simon Perdrix
CNRS, LORIA, Inria Mocqua, Université de Lorraine, F 54000 Nancy, France
simon.perdrix@loria.fr

July 7, 2020

Abstract

The prop formalism allows representation of processes with string diagrams and has been successfully applied in various areas such as quantum computing, electric circuits and control flow graphs. However, these graphical approaches suffer from scalability problems when it comes to writing large diagrams. A proposal to tackle this issue has been investigated for ZX-calculus using colored props. This paper extends the approach to any prop, making it a general tool for graphical languages manipulation.

There is a long list of graphical representations of various processes: Petri nets [4], control flow graphs [5], boolean circuits [18], quantum circuits [21], proof nets [13], *etc.* Most of them are unified into the prop formalism. A **prop** is a monoidal category with a free monoid of objects. Props were first introduced by Mac Lane in [19]. They are now used as a foundation for graphical reasoning through string diagrams. Each diagram is seen as a morphism in a prop. Props admit presentations by generators and equations [2].

Compared to usual symbolic manipulations, diagrams have the advantage of capturing some fondamental properties in a simple and intuitive way. In practice, most axioms are encoded into the topology making the notation lighter and easier to read. In the specific case of quantum computation, another advantage is the sometime radical compression allowing to represent matrices scaling exponentially with the number of qubits, by diagrams with a polynomial number of nodes. However, very large diagrams are difficult to draw, read and manipulate. Uniform diagrammatical proofs often use a lot of dots to suggest a repeating pattern or arbitrary large structures. The recent applications of these graphical methods and the need for verified proofs which cannot tolerate ellipsis lead to the development of new notations, allowing graphical languages to scale up.

Contributions. For every graphical language we provide a way to construct a *colored* graphical language where each wire is indexed by a size (its *color*), and can be interpreted as several wires side by side. We call this the *scalable construction*. We show that the scalable construction enjoys a universal property which gives us a natural way to extend the semantics of any (monochromatic) prop into semantics for the scalable version. We then describe how other props corresponding to substructures can be embedded into a scalable prop by introducing a *box construction*. This can be viewed as an abstraction of specific large graphical structures. Finally, we use previously known completeness results for graphical structures like monoids, bialgebras and Interacting Hopf Algebras, to provide a compressed way to deal with those structures in other graphical languages.

Related works. Our approach has been initiated in the special case of the ZX-calculus in [7], we extend this construction to any graphical language. The scalability problems for graphical languages has also motivated the introduction of !-boxes in [16]. Unlike our work, !-boxes do not provide a new graphical

language but a way to represent families of diagrams and to reason uniformly with them. In practice, scalable props provide a compact representation of large diagrams while !-boxes give a graphical description of how to construct a family of diagrams. Divider and gatherer generators similar to ours have been used in [20] to graphically define various matrix products. We share with this work our definition of direct sum of matrix arrows. The definition of our scaled generators matches what has been called monoidal multiplexing in [10], another work on the scalability problem. In fact this is the natural way to define the action of a diagram on many wires.

Structure of the paper. The first section sets up the prop framework to describe colored graphical languages. It provides the necessary definitions and results needed for the different scalable constructions. In the second section, the scalable constructions are described together with their main properties. The last section provides examples of large scale graphical structures that can benefit from our construction. We also give examples of concrete models for those structures, mainly taken from the ZX-, ZH- and ZW-calculi, introduced respectively in [11], [1] and [15].

1 Colored graphical languages

This section lays out the mathematical foundation for colored graphical reasoning. Nothing here is new, we just restate for graphical languages well known results of categorical universal alegebra.

1.1 Colored props

We work in the setting of colored props.

definition 1 (Colored prop)**.** *A **colored prop** is a small symmetric strict monoidal category* $\mathbf{P}$ *together with a set of **colors** C, such that the set of objects of* $\mathbf{P}$ *is freely spanned by the elements of C.*

We use the term **morphism** to denote an arrow of a colored prop as a category. From now on, we will use the term prop for a colored prop. A prop is **C-colored** when the set of colors is C and **monochromatic** when C is a singleton. The set of objects of a C-colored prop is C^*, the set of finite lists of elements of C. A list of n colors is denoted $\langle c_i \rangle_{i=1}^n$. We will write 0 for the empty list and $\boxtimes$ for the concatenation. This coincides with the monoidal structure in a C-colored prop. We have $0 \boxtimes \mathbf{a} = \mathbf{a} \boxtimes 0 = \mathbf{a}$ for all $\mathbf{a} \in C^*$. Given a color map $\phi : C \to C'$ we denote $\phi^* : C^* \to C'^*$ its extension to lists defined by $\phi^*(0) = 0$, $\phi^*(\mathbf{a} \boxtimes \mathbf{b}) = \phi^*(\mathbf{a}) \boxtimes \phi^*(\mathbf{b})$ and $\phi^*(\langle c \rangle) = \langle \phi(c) \rangle$.

A **prop morphism** is a symmetric strict monoidal functor mapping colors to colors. Formally, a prop morphism is a color map $\phi : C \to C'$ and a symmetric strict monoidal functor $F : \mathbf{P} \to \mathbf{P'}$ such that $F(\mathbf{a}) = \phi^*(\mathbf{a})$. **Prop** is the category of props and prop morphisms. This category is complete and co-complete. See [14] for a study of the category of props.

We will mainly work in subcategories of **Prop** where the set of colors is fixed. A **C-colored prop morphism** is a prop morphism between two C-colored props where $\phi = id_C$. For any set C of colors, **C-Prop** is the category of C-colored props and C-colored prop morphisms. **C-Prop** is a subcategory of **Prop** which is not full. By setting C to be a singleton we recover the category of monochromatic props and monochromatic prop morphisms that is often simply called **Prop** in the literature.

We will use the term **functor** when using functors in **SymCat** (the category of small symmetric monoidal category and symmetric monoidal functors), which in general are not prop morphisms.

We now mention some remarkable props. $\mathbf{1}_C$ is the codiscrete category over C^*, it is a terminal object in **C-Prop**. The terminal object for monochromatic props is denoted $\mathbf{N}$. $\mathbb{P}_C$ is the category of permutations over finite lists of colors in C. It is an initial object in **C-Prop**. The initial object for monochromatic props is denoted $\mathbb{P}$.

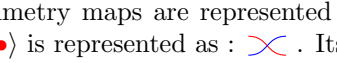
1.2 String diagrams

Colored props admit a nice graphical representation with colored string diagrams. Let $\mathbf{P}$ be a C-colored prop. The idea is that each color $c \in C$ corresponds to a color of wire. In this sub-section, we will actually

use various colors ($C = \{\bullet, \bullet, \bullet\}$) to represent the wires: the identity map $c \to c$ is represented as a wire of color c, e.g.  . Each morphism $\langle c_i \rangle_{i=1}^n \to \langle c'_j \rangle_{j=1}^m$ is represented as a diagram with n input wires colored according to the c_is and m output wires colored according to the c'_js. Here is an example for a map: $f : \langle \bullet, \bullet \rangle \to \langle \bullet, \bullet \rangle$:  . The empty object 0 and its identity map $id_0 : 0 \to 0$ correspond to the empty diagram . Diagrams of type $0 \to 0$ have no inputs nor outputs, we call them **scalars**.

In this representation the tensor product of two morphisms is represented by juxtaposition: . Thus the identity of the list $\langle \bullet, \bullet \rangle$, $id_{\langle \bullet, \bullet \rangle} = id_{\langle \bullet \rangle} \boxtimes id_{\langle \bullet \rangle}$, is represented as:  .

The composition of two morphisms is done by plugging the corresponding colored wires: .

The symmetry maps are represented by crossing the corresponding colored wires, for example $\sigma_{\langle \bullet, \bullet \rangle} : \langle \bullet, \bullet \rangle \to \langle \bullet, \bullet \rangle$ is represented as : . Its inverse is $\sigma_{\langle \bullet, \bullet \rangle} : \langle \bullet, \bullet \rangle \to \langle \bullet, \bullet \rangle$, graphically:  .

In this representation, the axioms of symmetric monoidal categories correspond to topological properties. The naturality of the symmetry corresponds to equations of the form:  . If two diagrams with colored wires are isomorphic then the corresponding morphisms are equal according to the axioms of C-colored props. See [23] for an overview of the different ways to draw categories.

1.3 Graphical languages

We describe props with their presentations by generators and equations. We call such an equational theory a graphical language. We start by defining colored signatures.

definition 2 (Signature)**.** *A colored* ***signature*** Σ *is a set of colors* C *together with a family of sets* $\Sigma[\mathbf{a}, \mathbf{b}]$ *indexed by* $\mathbf{a}, \mathbf{b} \in C^*$*. The elements of* $\Sigma[\mathbf{a}, \mathbf{b}]$ *are called* ***generators*** *of type* $\mathbf{a} \to \mathbf{b}$*. We write* $|\Sigma|$ *for* $\biguplus_{\mathbf{a},\mathbf{b}} \Sigma[\mathbf{a}, \mathbf{b}]$*, the set of all generators.*

A signature is a way to present a collection of morphisms of a colored prop that will be used later as building blocks, hence the name generators. We use the same terminology as props, in particular signature stands for colored signature and we say that a signature is monochromatic when the set of colors is a singleton. A way to present a signature is as a functor $C^* \times C^* \to \mathbf{Set}$, where $C^* \times C^*$ is a discrete category. Thus, there is a category of C-colored signatures C-**Sig** which is just another name for the functor category $\mathbf{Set}^{C^* \times C^*}$. The C**-colored signature morphisms** are natural transformations between signatures. In other words $\alpha : \Sigma \to \Sigma'$ is a familly of functions $\alpha_{\mathbf{a},\mathbf{b}} : \Sigma[\mathbf{a}, \mathbf{b}] \to \Sigma'[\mathbf{a}, \mathbf{b}]$.

The interest of this definition comes from the following result presented in [2].

theorem 1. *[2] Let* $U_C : C$-***Prop*** $\to C$-**Sig** *be the forgetful functor sending a* C*-colored prop* $\boldsymbol{P}$ *to the* C*-colored signature* $U_C(\boldsymbol{P})$ *such that for each* $a, b \in C^*$ $\Sigma[a, b] = \boldsymbol{P}[a, b]$*. U has a left adjoint* $F : C$-**Sig** $\to$ C-***Prop*** *and* C-***Prop*** *is equivalent to the Eilenberg-Moore category of the monad* $T \stackrel{def}{=} U \circ F$*. We denote* $\eta : 1 \Rightarrow T$ *the unit,* $\epsilon : F \circ U \Rightarrow 1$ *the counit and* $\mu : T^2 \Rightarrow T$ *the multiplication.*

The main interest of the theorem is to provide us with the free functor F sending a C-colored signature Σ to the free C-colored prop $F(\Sigma)$ spanned by this signature. Intuitively, a morphism in the free prop $F(\Sigma)$ is a string diagram built from generators in Σ linked by wires and swaps. From now on, by **diagram** we mean a morphism in $F(\Sigma)$. Such diagrams are identified up to the topological moves corresponding to the symmetric monoidal axioms. F can be used to define equational theories for props.

definition 3 (equation)**.** *An* ***equation*** *of type* $a \to b$ *with respect to a signature* Σ *is a pair* (f, g) *where* $f, g \in F(\Sigma)[a, b]$*.*

A set of equations over a signature Σ can be presented as a pair of signature morphisms $l, r : E \to UF(\Sigma)$. A graphical calculus is then formed by a signature and a set of equations over this signature.

definition 4 (C-colored graphical language)**.** *A C-colored graphical language is a tuple $\mathcal{L} \stackrel{def}{=} (\Sigma_\mathcal{L}, E_\mathcal{L}, l_\mathcal{L}, r_\mathcal{L})$ where $\Sigma_\mathcal{L}$ and $E_\mathcal{L}$ are C-colored signatures and $l_\mathcal{L}$ and $r_\mathcal{L}$ are C-colored signature morphisms of type $E_\mathcal{L} \to UF(\Sigma_\mathcal{L})$.*

To any graphical language $\mathcal{L} \stackrel{def}{=} (\Sigma_\mathcal{L}, E_\mathcal{L}, l_\mathcal{L}, r_\mathcal{L})$ corresponds an **underlying** prop $\mathbf{L}$, defined as the coequalizer: $FE_\mathcal{L} \underset{\epsilon_{F\Sigma_\mathcal{L}} \circ Fr_\mathcal{L}}{\overset{\epsilon_{F\Sigma_\mathcal{L}} \circ Fl_\mathcal{L}}{\rightrightarrows}} F\Sigma_\mathcal{L} \xrightarrow{\pi_\mathcal{L}} \mathbf{L}$.

It is defined only up to prop isomorphism. Given $f, g \in F(\Sigma_\mathcal{L})[n, m]$ we write $\mathcal{L} \vdash f = g$ iff $\pi_\mathcal{L}(f) = \pi_\mathcal{L}(g)$ in $\mathbf{L}$, this means that using the equations as local rewriting rules we can transform the diagram f into g.

We say that two graphical languages $\mathcal{L}$ and $\mathcal{Y}$ are **equipotent** iff $\mathbf{L} \simeq \mathbf{Y}$.

We define a **translation** between two graphical calculi as a signature morphism $\theta : \Sigma_\mathcal{L} \to UF(\Sigma_\mathcal{Y})$ satisfying the **soundness condition**: $\pi_\mathcal{Y} \circ (\epsilon_{F\Sigma_\mathcal{Y}} \circ F\theta) \circ (\epsilon_{F\Sigma_\mathcal{L}} \circ Fr_\mathcal{L}) = \pi_\mathcal{Y} \circ (\epsilon_{F\Sigma_\mathcal{Y}} \circ F\theta) \circ (\epsilon_{F\Sigma_\mathcal{L}} \circ Fl_\mathcal{L})$. This asserts that equivalent diagrams in $\mathcal{L}$ are sent to equivalent diagrams in $\mathcal{Y}$. The coequalizer property of $\mathbf{Y}$ gives a unique prop morphism $Bold(\theta)$ such that $Bold(\theta) \circ \pi_\mathcal{L} = \pi_\mathcal{Y} \circ (\epsilon_{F\Sigma_\mathcal{Y}} \circ F\theta)$. In fact, the soundness condition is equivalent to the existence of $Bold(\theta)$.

lemma 1. *There is a category **GL** of graphical languages and translations and an essentially surjective full functor $Bold : \mathbf{GL} \to \mathbf{Prop}$.*

A proof is given in A.1.

This functor allows us to define props and prop morphism using graphical languages and translations. Furthermore any prop and prop morphism admit such a description, but it is not unique.

C-**Prop** is a cocomplete category thus it has sums and coequalizers. These can be described with graphical languages:

definition 5 (Sum of graphical languages)**.** *The sum of two C-colored graphical languages $\mathcal{L}$ and $\mathcal{Y}$ is defined as $\mathcal{L} + \mathcal{Y} \stackrel{def}{=} (\Sigma_\mathcal{L} + \Sigma_\mathcal{Y}, E_\mathcal{L} + E_\mathcal{Y}, l_{\mathcal{L}+\mathcal{Y}}, r_{\mathcal{L}+\mathcal{Y}})$, where $l_{\mathcal{L}+\mathcal{Y}} = [UF(\iota_{\Sigma_\mathcal{L}}) \circ l_\mathcal{L}, UF(\iota_{\Sigma_\mathcal{L}}) \circ l_\mathcal{Y}]$ and $r_{\mathcal{L}+\mathcal{Y}} = [UF(\iota_{\Sigma_\mathcal{L}}) \circ r_\mathcal{L}, UF(\iota_{\Sigma_\mathcal{L}}) \circ r_\mathcal{Y}]$.*

The sum of two graphical languages has the generators and equations of both languages.

lemma 2. *$Bold(\mathcal{L} + \mathcal{Y})$ is a coproduct $\mathbf{L} + \mathbf{Y}$ in C-**Prop**.*

A proof is given in A.1.

Usually when we build new graphical languages we take the sum of other graphical languages and then add more equations.

definition 6 (Quotient of a graphical language by equations)**.** *Given a C-colored graphical languages $\mathcal{L}$ and a set of equations (E, l, r) over $\Sigma_\mathcal{L}$, the quotient of $\mathcal{L}$ by E is defined as $\mathcal{L}/_E \stackrel{def}{=} (\Sigma_\mathcal{L}, E_\mathcal{L} + E, [l_\mathcal{L}, l], [r_\mathcal{L}, r])$.*

lemma 3. *$Bold(\mathcal{L}/_E)$ is a coequalizer of $\pi_\mathcal{L} \circ \epsilon_{F\Sigma_\mathcal{L}} \circ Fl$ and $\pi_\mathcal{L} \circ \epsilon_{F\Sigma_\mathcal{L}} \circ Fr$ in C-**Prop**.*

A proof is given in A.1.

The semantics of a graphical language is given by interpretation functors.

definition 7. *An **interpretation** of a graphical language $\mathcal{L}$ is a functor $[\![\_]\!] : \mathbf{L} \to \mathbf{C}$. A language is said **complete**, respectively **universal**, for the category $\mathbf{C}$ if there exists an interpretation which is full, respectively faithful.*

A natural question, given a graphical language, is to find an interpretaion for which the language is universal and complete. Such examples will be given in section 3. We are now ready to introduce the scalable construction.

2 The scalable construction

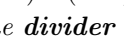
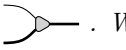
From now on, our set of colors will always be $\mathbb{N}_0 := \mathbb{N} \setminus \{0\}$. In this setting we will say **size** instead of color. We will use string diagrams to represent morphisms. A wire of size 1 is said **simple** and a wire of size $n > 1$ is said **big**. To simplify the notations, a thin wire will always be simple when a bold one without label can be of any size. We will only use size labels when absolutely necessary.

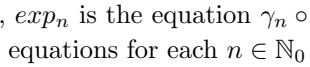
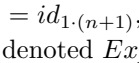
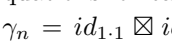
The objects of an $\mathbb{N}_0$-colored prop are lists of positive integers. The list with k times the integer n: $\langle n \rangle_{i=1}^{k}$ is denoted $k \cdot n$. We have $0 \cdot n = k \cdot 0 = 0$ but $k \cdot n \neq n \cdot k$ and in particular $k \cdot 1 \neq 1 \cdot k$. In fact $k \cdot 1$ corresponds to k simple wires and $1 \cdot k$ to one big wire of size k. Given a family of wires of arbitrary size, which corresponds to an arbitrary object in an $\mathbb{N}_0$-colored prop, we define a notion of global size.

definition 8. *Given an $\mathbb{N}_0$-colored prop $\boldsymbol{P}$, the **global size** functor $[\_] : \boldsymbol{P} \to \mathbf{N}$ is the unique functor satisfying: $[0] = 0$ and $[1 \cdot n] = n$.*

Intuitively big wires of size k are ribbon cables, representing k simple wires together, the global size functor just counts the overall number of simple wires. This intuition is made precise by the wire calculus.

2.1 The wire calculus

definition 9 ($\mathcal{D}$ and $\mathcal{G}$)**.** *The graphical languages $\mathcal{D}$ and $\mathcal{G}$ (for dividers and gatherers) are respectively freely generated by the signatures $\Sigma_{\mathcal{D}}[1 \cdot (n+1), (1 \cdot 1) \boxtimes (1 \cdot n)] \overset{def}{=} \{\delta_n\}$ and $\Sigma_{\mathcal{G}}[(1 \cdot 1) \boxtimes (1 \cdot n), 1 \cdot (n+1)] \overset{def}{=} \{\gamma_n\}$ for all $n \in \mathbb{N}_0$. The generator δ_n is called the **divider** of size n and is depicted:*  *. The generator γ_n is called the **gatherer** of size n and is depicted:*  *. We take the conventions: $\delta_0 \overset{def}{=} id_1$ and $\gamma_0 \overset{def}{=} id_1$.*

We have $\mathbf{G} \simeq \mathbf{D}^{op}$. We then define how those props interact in the spirit of [17]. The **expansion equation** of size n, exp_n is the equation $\gamma_n \circ \delta_n = id_{1\cdot(n+1)}$, pictorially:  $\overset{exp}{=}$  . The set of all expansion equations for each $n \in \mathbb{N}_0$ is denoted Exp. The **elimination equation** of size n, $elim_n$ is the equation $\delta_n \circ \gamma_n = id_{1\cdot 1} \boxtimes id_{1\cdot n}$, pictorially:  $\overset{exp}{=}$  . The set of all elimination equations for each $n \in \mathbb{N}_0$ is denoted $Elim$.

The convention for δ_0 and γ_0 makes exp_0 and $elim_0$ trivially true.

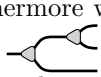
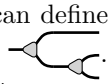
definition 10 (The wire calculus $\mathcal{W}$)**.** *The $\mathbb{N}_0$-colored graphical language $\mathcal{W}$ is defined as $\mathcal{W} \overset{def}{=} (\mathcal{D} + \mathcal{G}) \Big/ (Elim, Exp)$.*

In $\mathcal{W}$, the role of dividers and gatherers is perfectly symmetric, thus we have $\mathbf{W} \simeq \mathbf{W}^{op}$. $\mathbf{W}$ is a groupoid, in fact the elimination and expansion rules exactly state that the generators are invertible. We also note that the generators preserve the global size so there is no morphism of type $\mathbf{a} \to \mathbf{b}$ when $[\mathbf{a}] \neq [\mathbf{b}]$. In fact we can go further: when $[\mathbf{a}] = [\mathbf{b}]$, the morphisms of type $\mathbf{a} \to \mathbf{b}$ are the permutations of $[\mathbf{a}]$.

theorem 2 (Rewiring theorem)**.** ***W** is a full subcategory of the permutation category $\mathbb{P}_{\mathbb{N}_0}$, satisfying $[\mathbf{a}] \neq [\mathbf{b}] \Rightarrow \boldsymbol{W}[\mathbf{a}, \mathbf{b}] = \emptyset$.*

A proof is given in A.2.

This gives us a clear understanding of what $\mathbf{W}$ looks like as a category. The rewiring theorem works as a coherence result in the sense that any equation is true up to permutations as soon as the types match. This gives us total freedom to rewire the way we want.

The way the dividers and gatherers are defined, taking the wires one by one, is useful to come up with a normal form but is still quite restrictive. The rewiring theorem allows us to unambiguously generalize dividers and gatherers to any wire size. Furthermore we can define a divider with an arbitrary number of outputs since the associativity equation holds:  $=$  . We define inductively $\Delta_n : 1 \cdot n \to n \cdot 1$ by $\Delta_0 = id_0$, and $\Delta_{n+1} = (id_1 \otimes \Delta_n) \circ \delta_n$. The diagram Δ_n is a sequence of n dividers of decreasing size. For any object $\mathbf{a} = \langle n_i \rangle_{i=1}^{k}$ we define $\Delta_{\mathbf{a}} \overset{def}{=} \bigotimes_i \Delta_{n_i}$. Dually, we define $\Gamma_m : m \cdot 1 \to 1 \cdot m$ by $\Gamma_0 = id_0$, and

$\Gamma_{m+1} = \gamma_m \circ (id_1 \otimes \Gamma_m)$, which is a sequence of m gatherers of increasing size. For any object $\mathbf{b} = \langle m_i \rangle_{i=1}^k$ we define $\Gamma_{\mathbf{b}} \stackrel{def}{=} \bigotimes_i \Gamma_{m_i}$.

We now proceed to making a monochromatic prop interacting with dividers and gatherers through the scalable construction.

2.2 The scalable construction

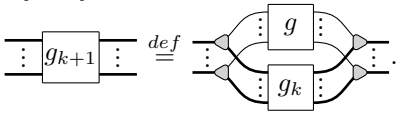
Intuitively, the scalable construction is the free embedding of a monochromatic graphical language into the simple wires of **W**.

definition 11 ($\mathcal{SL}$)**.** *Given a monochromatic graphical language $\mathcal{L}$, we define an $\mathbb{N}_0$-colored graphical language $\mathcal{L}^{\mathbb{N}_0}$ by:*

$$\Sigma_{\mathcal{L}^{\mathbb{N}_0}}[a,b] \stackrel{def}{=} \begin{cases} \Sigma_{\mathcal{L}}[n,m] & \text{if } (a,b)=(n\cdot 1, m\cdot 1) \\ \emptyset & \text{otherwise} \end{cases} \qquad E_{\mathcal{L}^{\mathbb{N}_0}}[a,b] \stackrel{def}{=} \begin{cases} E_{\mathcal{L}}[n,m] & \text{if } (a,b)=(n\cdot 1, m\cdot 1) \\ \emptyset & \text{otherwise} \end{cases}$$

*The **scalable** graphical calculus $\mathcal{SL}$ is defined as: $\mathcal{SL} \stackrel{def}{=} \mathcal{L}^{\mathbb{N}_0} + \mathcal{W}$. The underlying prop Bold($\mathcal{SL}$) is denoted $\boldsymbol{\mathcal{SL}}$.*

The scalable construction was first introduced in [9] to allow a compact representation of large diagrams with an identifiable large scale structure. In fact, starting from $\mathcal{SL}$ we can add syntactic sugar to handle large scale graphical rewriting.

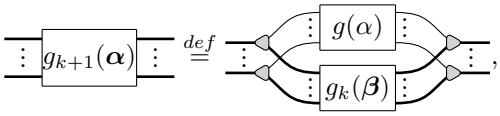
Given a diagram $g \in \mathcal{L}[n,m]$, its scaled version of size k is a diagram $g_k \in \mathcal{SL}[k \cdot n, k \cdot m]$ inductively defined by: $g_1 \stackrel{def}{=} g$ and 

. This transformation has been called monoidal multiplexing in [10]. Notice that these structures require a way to cross wires, here we have a symmetry but a braiding would also work.

lemma 4. *For every k, there is a functor $S_k : \boldsymbol{L} \to \boldsymbol{\mathcal{SL}}$, such that $S_k(1) = 1 \cdot k$ and $S_k(\pi_{\mathcal{L}}(g)) = \pi_{\mathcal{SL}}(g_k)$.*

A proof is given in A.2.

These functors ensure that any equation between diagrams still holds at large scale where one application of the scaled rule is in fact hiding k parallel applications of the original rule.

We can go further, if $(g(\alpha) : \mathbf{a} \to \mathbf{b})_{\alpha \in A}$ is a family of diagrams indexed by some parameter $\alpha \in A$ then we can index the scaled version by an element $\boldsymbol{\alpha} \in A^k$. This is defined inductively by $g_1(\langle \alpha \rangle) \stackrel{def}{=} g(\alpha)$ and:


, with $\boldsymbol{\alpha} = \langle \alpha \rangle \boxtimes \boldsymbol{\beta}$ and $\boldsymbol{\beta} \in A^k$.

Notice that this is how scale spiders are defined in the scalable ZX-calculus [7]. We see here that this general construction applies without difficulties to the other kinds of indexed spiders one can find in ZW or ZH calculi, and to the boxes indexed by numbers in [6]. The associated scaled rule, if any, should be *a priori* defined inthe same way. For example, in the ZX-calculus, the phases α and β of two spiders add up when they fuse, so the lists of phases $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ add up pointwisely when scaled spiders fuse.

This construction applies naturally to generators but can be applied to any diagrams that we want to see acting at a large scale.

We can refine the global size functor into a functor from $\mathcal{SL}$ to **L** which forgets dividers and gatherers.

definition 12 (Wire stripper)**.** *Given a monochromatic graphical language $\mathcal{L}$, the **wire stripper** functor $\{\_\} : \boldsymbol{\mathcal{SL}} \to \boldsymbol{L}$ is defined on colors by $\{\mathbf{a}\} = [\mathbf{a}]$ and on morphisms by $\{\pi_{\mathcal{SL}}(\delta_k)\} = id_k$, $\{\pi_{\mathcal{SL}}(\gamma_k)\} = id_k$ and $\{\pi_{\mathcal{SL}}(x)\} = \pi_{\mathcal{L}}(x)$ for all generators $x \in \Sigma_{\mathcal{L}}$.*

lemma 5. *For every morphism $\omega \in \boldsymbol{L}[n,m]$, we have $\{S_1(\omega)\} = \omega$.*

A proof is given in A.2.

All the properties of $\mathcal{SL}$ follow from a structure theorem which can be seen as an extension to $\mathcal{SL}$ of the rewiring theorem.

theorem 3 (Structure of $\mathcal{S}\mathbf{L}$)**.** *For every $\omega \in \mathcal{SL}[\mathbf{a}, \mathbf{b}]$ we have $\mathcal{SL} \vdash \omega = \Gamma_{\mathbf{b}} \circ S_1(\{\omega\}) \circ \Delta_{\mathbf{a}}$.*

A proof is given in A.2.

Starting with a graphical language for permutations $\mathbb{P}$ we have $\mathcal{S}\mathbb{P} = \mathbf{W}$ and then we recover the rewiring theorem. From this result it follows that the scalable construction enjoys a universal property:

lemma 6 (Universal property of $\mathcal{S}\mathbf{L}$)**.** *The following diagram is a pullback square:*

$$\begin{array}{ccc} \mathcal{S}\mathbf{L} & \xrightarrow{\{\_\}} & \mathbf{L} \\ {\scriptstyle !}\downarrow & \lrcorner & \downarrow{\scriptstyle !} \\ \mathbf{1}_{\mathbb{N}_0} & \xrightarrow[[\_]]{} & \mathbf{N} \end{array}$$

A proof is given in A.2.

The universal property allows to lift interpretation functors.

lemma 7 (Scaled interpretation)**.** *Given a monochromatic prop $\boldsymbol{C}$ and an interpretation $[\![\_]\!] : \boldsymbol{L} \to \boldsymbol{C}$, let $\boldsymbol{C}_p$ the be the symmetric strict monoidal category whose objects are pairs (n, p_n) where p_n is a partition of n, and such that $\boldsymbol{C}_p[(n, p_n), (m, p_m)] = \boldsymbol{C}[n, m]$. $\boldsymbol{C}_p$ is an $\mathbb{N}_0$-colored prop and there is a unique $\mathbb{N}_0$-colored prop morphism $[\![\_]\!]_p : \mathcal{S}\boldsymbol{L} \to \boldsymbol{C}_p$ such that $[\_] \circ [\![\_]\!]_p = [\![\_]\!] \circ \{\_\}$. We call it the **scaled interpretation** and it is faithful iff $[\![\_]\!]$ is faithful.*

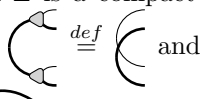
A proof is given in A.2.

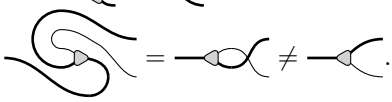
These results together point out that the scalable construction is just a tool allowing diagrammatical manipulation and is completely orthogonal to the original language. In fact, we even have an equivalence of categories.

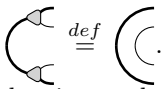
lemma 8. *In **SymMonCat** we have $\boldsymbol{L} \simeq \mathcal{S}\boldsymbol{L}$.*

A proof is given in A.2.

We expect most of the properties of $\mathbf{L}$ to be reflected in $\mathcal{S}\mathbf{L}$. Here are some specific examples. If $\mathbf{L}$ is a dagger category then $\mathcal{S}\mathbf{L}$ inherits this structure by setting ${\delta_k}^\dagger \stackrel{def}{=} \gamma_k$. Then we have ${\gamma_k}^\dagger = \delta_k$ and the expansion and elimination equations state that dividers and gatherers are unitary maps.

If $\mathbf{L}$ is a compact closed category then so is $\mathcal{S}\mathbf{L}$. Using the scaled version of the cups and caps, we have  and then ${\gamma_k}^t = \delta_k$. However, note that some intuitive topological moves do not hold:



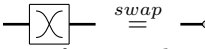
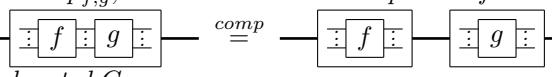
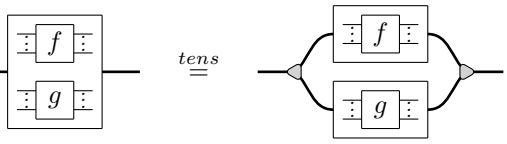
Remark: Another possibility is to take  . Then we recover the topology but we loose the correspondance between equations on simple wires and their scaled version.

3 The box construction

In this section, we focus on some fundamental graphical languages. We use the associated completeness results to compress the corresponding diagrams with a box construction. We consider a way to construct an $\mathbb{N}_0$-colored graphical language from a monochromatic one: the **box construction**. The idea is to blackbox a monochromatic prop into a large scale graphical language.

definition 13 ($Box_{\mathbf{P}}$). *Given a monochromatic prop $\boldsymbol{P}$, let $\boldsymbol{P}^{\mathbb{N}_0}$ be the $\mathbb{N}_0$-colored graphical language defined as* $\Sigma_{\boldsymbol{P}^{\mathbb{N}_0}}[\mathbf{a},\mathbf{b}] \stackrel{def}{=} \begin{cases} U\boldsymbol{P}[n,m] & \text{when } (\mathbf{a},\mathbf{b}) = (1\cdot n, 1\cdot m) \\ \emptyset & \text{otherwise} \end{cases}$*, and no equation. For every morphism $f : n \to m$ of $\boldsymbol{P}$ the corresponding generator in $\boldsymbol{P}^{\mathbb{N}_0}$ is denoted $\square_f$.*

*The **box graphical language** is defined as* $Box_{\boldsymbol{P}} \stackrel{def}{=} \left(\boldsymbol{P}^{\mathbb{N}_0} + \mathcal{W}\right) \Big/ (Swap, Comp, Tens)$*, where swap, Comp, Tens are the following equations:*

- *The **Swap equation** swap is: $\square_{\sigma_{1,1}} = \gamma_2 \circ \sigma_{1,1} \circ \delta_2$, pictorially:*  *.*
- *The **composition equation** $comp_{f,g}$, associated to two morphisms $f : n \to k$ and $g : k \to m$ of $\boldsymbol{P}$, is $\square_{g\circ f} = \square_g \circ \square_f$, pictorially:*  *. The set of all composition equations for every f and g is denoted $Comp$.*
- *The **tensor equation** $tens_{f,g}$, associated to two morphisms $f : n \to m$ and $g : k \to l$ of $\boldsymbol{P}$, is $\square_{f\otimes g} = \gamma_{m+l} \circ (\square_f \boxtimes \square_g) \circ \delta_{n+k}$, pictorially:*  *. The set of all tensor equations for every f and g is denoted $Tens$.*

The $\mathbb{N}_0$-colored prop associated to the **box graphical language** $Box_{\mathbf{P}}$ is denoted $Box\mathbf{P}$. Notice that the box graphical language has one generator for each morphism in $\mathbf{P}$.

From this definition follows directly the existence of a functor $B : \mathbf{P} \to Box\mathbf{P}$ defined on each morphism $f : n \to m$ by $B(f) = \Delta_m \circ \pi_{Box_{\mathbf{P}}}(\square_f) \circ \Gamma_n$. The equations of $Box_{\mathbf{P}}$ state exactly that B is a symmetric strict monoidal functor. We also have a functor $O : Box\mathbf{P} \to \mathbf{P}$ defined on generators by $O(\pi_{Box_{\mathbf{P}}}(\square_f)) = f$. We have $O \circ B = id_{\mathbf{P}}$.

As with scalable construction, we also have a structure theorem:

theorem 4 (Structure of $Box\mathbf{P}$). *For each $\omega \in Box(\boldsymbol{P})[\mathbf{a},\mathbf{b}]$ we have $Box(\boldsymbol{P}) \vdash \omega = \Gamma_{\mathbf{b}} \circ B(O(\omega)) \circ \Delta_{\mathbf{a}}$.*

A proof is given in A.3.

From this given a monochromatic graphical language $\mathcal{L}$ we can define a functor $Unbox : Box\mathbf{L} \to \mathcal{S}\mathbf{L}$ by $Unbox(\omega) = \Gamma_{\mathbf{b}} \circ S_1(O(\omega)) \circ \Delta_{\mathbf{a}}$.

lemma 9. *$Unbox$ is an equivalence of categories.*

A proof is given in A.3.

So $Box\mathbf{L} \simeq \mathcal{S}\mathbf{L}$, as a consequence, the two constructions are essentially the same.

We will mostly use the box construction on substructures of $\mathcal{L}$ to obtain box generators inside of $\mathcal{L}$. Of course they are expressible using the usual generators, but they are very useful for compressing diagrams and speed up graphical computations. We now give several examples of various kinds of boxes and some of their interactions with scaled generators.

3.1 Symmetries and permutations

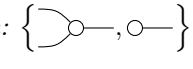
We work in the setting of props so the graphical language of permutation is the free graphical language with no signature nor equations. Making an exception, we describe here the language in the setting of pros in order to start with a familiar example.

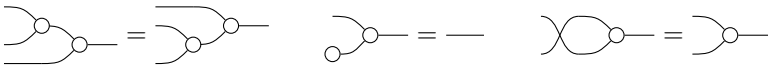
definition 14 (permutations). *The monochromatic graphical language $\mathcal{P}$ has signature:*  *and equations:* *and* .

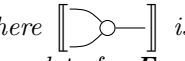
lemma 10. *The interpretation* $[\![$$]\!] = (1,2)(2,1)$ *makes $\mathcal{P}$ complete for the prop of permutations $\mathbb{P}$ where each morphism $f : n \to n$ is a permutation of $\{1,\dots,n\}$. Composition is the composition of permutations and the tensor product is the disjoint union.*

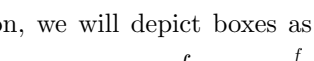
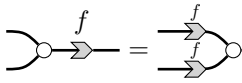
Given any monochromatic prop $\mathbf{P}$ there is a unique prop morphism $! : \mathbb{P} \to \mathbf{P}$ from the prop of permutations. Thus in any scalable prop we can use permutation boxes from $Box\mathbb{P}$ without ambiguity.

3.2 Monoid and functions

definition 15 (Commutative monoid). *The monochromatic graphical language $\mathcal{M}$ has signature:*  *and equations:*

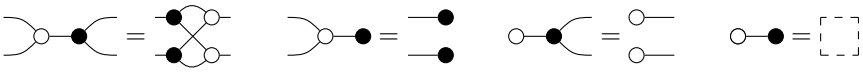


lemma 11. ***Fun*** *is the prop of functions where each morphism $f : n \to m$ is a function from $\{1,\dots,n\}$ to $\{1,\dots,m\}$. Composition is the composition of functions and the tensor product is the disjoint union. The interpretation where*  *is the unique function $\{1,2\} \to \{1\}$ and*  *is the unique function $\emptyset \to \{1\}$, makes $\mathcal{M}$ complete for* ***Fun****.*

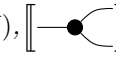
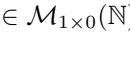
From now on, we will depict boxes as arrows to fit the notation of [7]:  . Every function arrow satisfies:  and .

3.3 Bialgebras and matrices

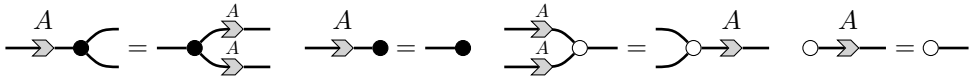
definition 16 (Commutative bialgebra). *The monochromatic graphical language $\mathcal{B}$ is defined as $\mathcal{M} + \mathcal{M}^{op}$ quotiented by the equations:*

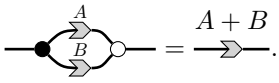


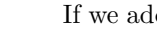
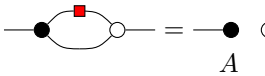
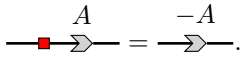
where the generators of $\mathcal{M}$ are in white and those of $\mathcal{M}^{op}$ in black.

lemma 12. *[22] $\mathcal{M}(\mathbb{N})$ is the prop of integer matrices where each morphism $f : n \to m$ is a matrix in $\mathcal{M}_{m\times n}(\mathbb{N})$. Composition is the matrix product and the tensor product is the direct sum. The interpretation* $[\![$  $]\!] = \begin{pmatrix}1 & 1\end{pmatrix}$, $[\![$  $]\!] = () \in \mathcal{M}_{1\times 0}(\mathbb{N})$, $[\![$  $]\!] = \begin{pmatrix}1\\1\end{pmatrix}$ *and* $[\![$$]\!] = () \in \mathcal{M}_{0\times 1}(\mathbb{N})$ *makes $\mathcal{B}$ complete for $\mathcal{M}(\mathbb{N})$.*

A matrix arrow indexed by $A : n \to m$ corresponds to a bipartite multigraph between n black vertices and m white vertices. A is nothing but the biadjacency matrix of this multigraph. Thus, in the presence of bialgebra, the box construction allows to compress a bipartite sub-diagram into a single matrix arrow. The properties of matrix arrows generalize the ones of function boxes. The following equations hold for any matrix A:

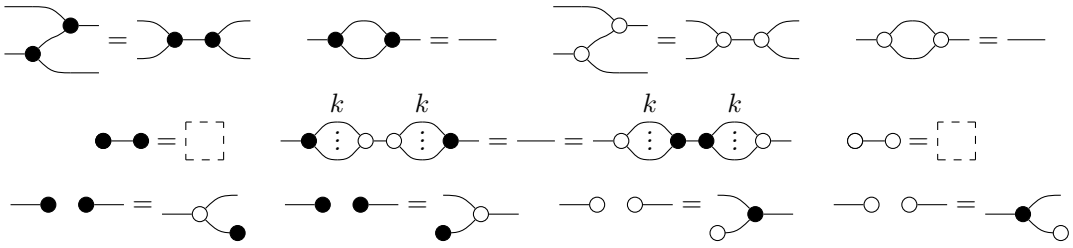


Furthermore:  .

If we add to $\mathcal{B}$ the generator  and the equation  we obtain the graphical language $\mathcal{H}$ of Hopf algebras. $\mathcal{H}$ is complete for matrices over $\mathbb{Z}$ with  .

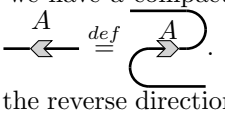
3.4 Interacting Hopf algebras and linear relations

definition 17 (Interacting Hopf algebras)**.** *The monochromatic graphical language $\mathcal{IH}$ is defined as $\mathcal{B}+\mathcal{B}^{op}$ quotiented by the equations:*

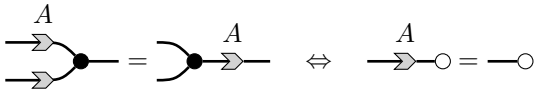


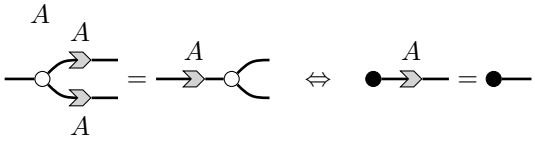
The generators of $\mathcal{B}$ are depicted as before and those of $\mathcal{B}^{op}$ are depicted by exchanging black and white. The number k of parallel wires must be at least 2.

lemma 13. *[6]* ***LinRel*** *is the prop of linear relation where each morphism $f : n \to m$ is a linear subspace of $\mathbb{Q}^{n+m}$. Composition is composition of relation and the tensor product is direct sum. The interpretation:* $[\![$  $]\!] = \{(x, y, z), x + y = z\}$, $[\![$  $]\!] = \{0\} \times \{0\}$, $[\![$  $]\!] = \{(x, y, z), x = y + z\}$, $[\![$  $]\!] = \{0\} \times \{0\}$, $[\![$  $]\!] = \{(x, x, x), x \in \mathbb{Q}\}$, $[\![$ $]\!] = \{0\} \times \mathbb{Q}$, $[\![$ $]\!] = \{(x, x, x), x \in \mathbb{Q}\}$ *and* $[\![$  $]\!] = \mathbb{Q} \times \{0\}$ *makes $\mathcal{IH}$ complete for* ***LinRel****.*

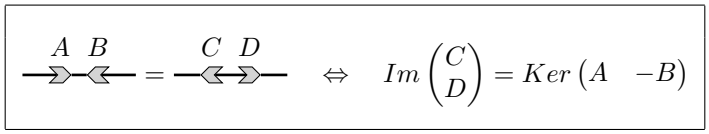
We can interpret matrices in $\mathcal{IH}$ and thus have matrix arrows. Moreover we have a compact structure on simple wires given by  and  . We define backward matrix as  .

Backward matrix arrows have the same properties as matrix arrows but in the reverse direction. We also have:

A is injective $\Leftrightarrow$ 

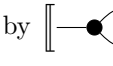
A is surjective $\Leftrightarrow$ 

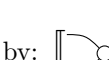
In practice we will not use linear relation boxes but only matrix arrows since any linear relation factorizes into matrix arrows. In fact, all the equations and properties given in this section so far can be summed up in the powerful formula from [24]:



4 Examples in graphical languages

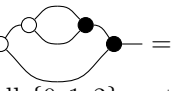
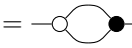
In practice, when we identify a substructure in a graphical language we can use the scalable technics to manipulate boxes. However, in general those are not free and we can have extra equations between boxes. Moreover, sometimes the structures are a little different than expected and thus some adjustments need to be made on the properties of boxes. We illustrate this by considering three concrete examples of bialgebras from categorical quantum mechanics. They appear respectively in the ZH, ZW and ZX-calculus. Notice that we use the definitions of these bialgebras given in [8] which are essentially equivalent, but may slightly differ from the original ones. Each of the three graphical languages has an interpretation which make it complete for the prop of qubits where each morphism $f : n \to m$ is a matrix in $\mathcal{M}_{2^m,2^n}(\mathbb{C})$. The composition is the matrix product and the tensor is the tensor product of matrices. A basis of $\mathbb{C}^{2^n}$ is denoted by the $|x\rangle$ where x is a binary word of size n. e_i is the binary word with 0 everywhere except in the ith coordinate where it is 1.

We recall that a bialgebra corresponds to matrices over the semiring $(\mathbb{N}, +, \times)$. The three bialgebras share the same comonoid defined by $[\![$  $]\!] = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$ and $[\![$  $]\!] = \begin{pmatrix} 1 & 1 \end{pmatrix}$.

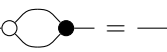
The **ZH bialgebra** is defined by: $[\![$  $]\!] = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ and $[\![$  $]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

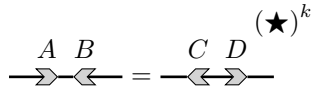
We have  = —, which essentially means "2=1". As a consequence, we are working with matrices over the semiring $(\mathbb{N}, +, \times) \Big/ (2 = 1)$. This is exactly the boolean semiring $(\vee, \wedge)$. There is no more quotienting since $A|e_j\rangle = |A_j\rangle$ where A_j is the jth column of A. All $\{0, 1\}$-matrix arrows are different.

The **ZW bialgebra** is defined by $[\![$  $]\!] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$ and $[\![$  $]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

We have  =  . Thus, we are working with matrices over the semiring $(\mathbb{N}, +, \times) \Big/ (2 = 3)$. However, not all $\{0, 1, 2\}$-matrices have a distinct interpretation. If A is a $\{0, 1\}$-matrix then $A|e_j\rangle = |A_j\rangle$, but if A has a 2 in the jth column then for all x such that $x_j = 1$ we have $A|x\rangle = 0$. We cannot distinguish the coefficients in a column with a 2. So the arrows corresponds to matrices with only $\{0, 1\}$-coefficients except in some columns which are full of 2s.

The **ZX bialgebra** is defined by $[\![$  $]\!] = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$ and $[\![$  $]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

We have  =  . We are working with matrices over the semiring $(\mathbb{N}, +, \times) \Big/ (2 = 0)$. This is the field $\mathbb{F}_2$. There is no more quotienting since $A|e_j\rangle = |A_j\rangle$. All $\{0, 1\}$-matrices have a distinct interpretation. As the presence of a field suggests we can extend ZX into a **scaled** interacting Hopf algebras. This means the equational theory of $\mathcal{IH}$ holds up to some scalars. We can still work with linear relations over $\mathbb{F}_2$ but the main equations needs to be renormalized as follow:



$$\Leftrightarrow \quad Im \begin{pmatrix} C \\ D \end{pmatrix} = Ker \begin{pmatrix} A & B \end{pmatrix}$$

Where $k \stackrel{def}{=} dim\left(Ker\begin{pmatrix} C \\ D \end{pmatrix}\right)$ and $[\![\bigstar]\!] = \frac{1}{\sqrt{2}}$. This equation subsumes most of the properties of the $\mathcal{S}ZX$-calculus of [7].

References

[1] Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. In Peter Selinger and Giulio Chiribella, editors, Proceedings of the 15th International Conference on *Quantum Physics and Logic,* Halifax, Canada, 3-7th June 2018, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 23–42. Open Publishing Association, 2019. doi:10.4204/EPTCS.287.2.

[2] John C Baez, Brandon Coya, and Franciscus Rebro. Props in network theory. *Theory and Applications of Categories*, 33(25):727–783, 2018.

[3] Michael Barr and Charles Wells. Toposes, theories, and triples. *Reprints in Theory and Applications of Categories*, 12:1–287, 2005.

[4] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–28, 2019.

[5] Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. Full abstraction for signal flow graphs. *ACM SIGPLAN Notices*, 50(1):515–526, 2015.

[6] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Interacting hopf algebras. *Journal of Pure and Applied Algebra*, 221(1):144–184, 2017.

[7] Titouan Carette, Dominic Horsman, and Simon Perdrix. Szx-calculus: Scalable graphical quantum reasoning. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[8] Titouan Carette and Emmanuel Jeandel. A recipe for quantum graphical languages. *47th International Colloquium on Automata, Languages and Programming*, 2020.

[9] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, and Dominic Horsman. Graphical structures for design and verification of quantum error correction. *arXiv preprint arXiv:1611.08012*, 2016.

[10] Apiwat Chantawibul and Paweł Sobociński. Monoidal multiplexing. In *International Colloquium on Theoretical Aspects of Computing*, pages 116–131. Springer, 2018.

[11] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.

[12] Pierre-Louis Curien. The joy of string diagrams. In *International Workshop on Computer Science Logic*, pages 15–22. Springer, 2008.

[13] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. In *Logic and Algebra*, pages 97–124. Routledge, 2017.

[14] Philip Hackney and Marcy Robertson. On the category of props. *Applied Categorical Structures*, 23(4):543–573, 2015.

[15] Amar Hadzihasanovic. Zw calculi: diagrammatic languages for pure-state quantum computing. *Logic and Applications LAP 2018*, page 13, 2018.

[16] Aleks Kissinger and David Quick. Tensors,!-graphs, and non-commutative quantum structures. *New Generation Computing*, 34(1-2):87–123, 2016.

[17] Stephen Lack. Composing props. *Theory and Applications of Categories*, 13(9):147–163, 2004.

[18] Yves Lafont. Towards an algebraic theory of boolean circuits. *Journal of Pure and Applied Algebra*, 184(2-3):257–310, 2003.

[19] Saunders MacLane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71(1):40–106, 1965.

[20] Filippo M Miatto. Graphical calculus for products and convolutions. *arXiv preprint arXiv:1903.01366*, 2019.

[21] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

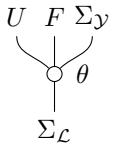
[22] Teimuraz Pirashvili. On the *prop* corresponding to bialgebras. *Cahiers de topologie et géométrie différentielle catégoriques*, 43(3):221–239, 2002.

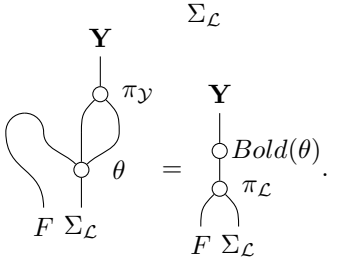
[23] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

[24] Fabio Zanasi. Interacting hopf algebras: the theory of linear systems. *PhD thesis, arXiv preprint arXiv:1805.03032*, 2018.

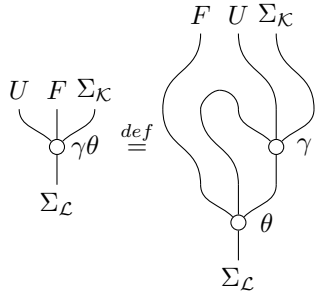
A Proofs

A.1 Proofs for section 1

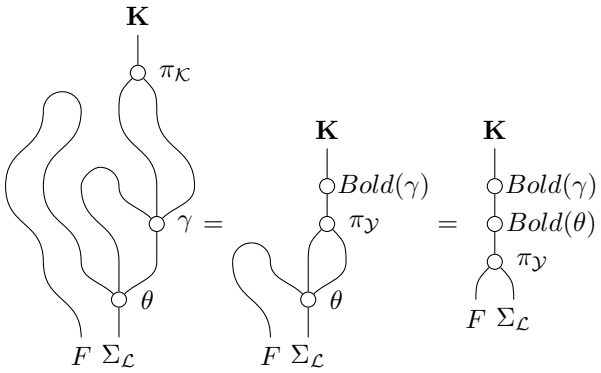
Proof of Lemma 1. We will use the string diagrams notation for natural transformations. See [12]. A translation is pictured: 

. The soundness condition states that there is a prop morphism $Bold(\theta)$ such that: 

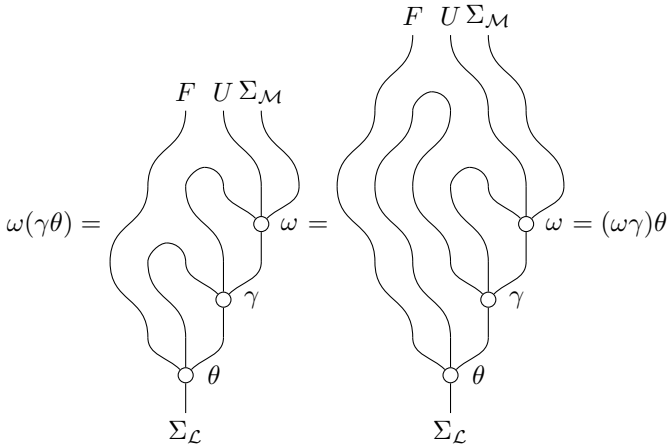
. The composition between to translations $\theta : \mathcal{L} \to \mathcal{Y}$ and $\gamma : \mathcal{Y} \to \mathcal{K}$ is defined as $\gamma\theta \stackrel{def}{=} \mu_{\Sigma_\mathcal{K}} \circ T\gamma \circ \theta$, pictorially:



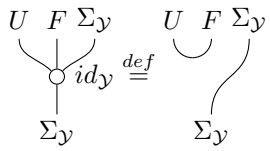
We need to check the soundness condition for the composition. We have:



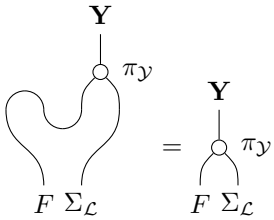
So the soundness condition is satisfied with $Bold(\gamma\theta) = Bold(\gamma) \circ Bold(\theta)$.
Given three translations $\theta : \mathcal{L} \to \mathcal{Y}$, $\gamma : \mathcal{Y} \to \mathcal{K}$ and $\omega : \mathcal{K} \to \mathcal{M}$ we have:



So the composition is associative.
The unit is defined as $id_{\mathcal{Y}} \stackrel{def}{=} \eta_{\Sigma_{\mathcal{Y}}}$, pictorially:

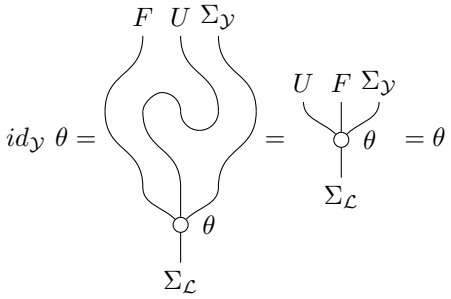


The soundness condition is satisfied:

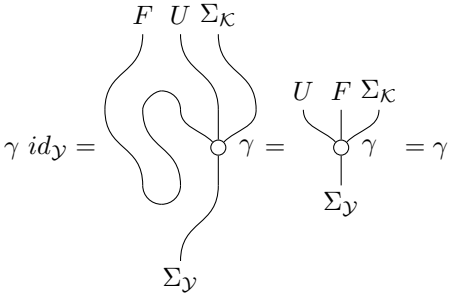


and we have $Bold(id_{\mathcal{Y}}) = id_{\mathbf{Y}}$.

We have:



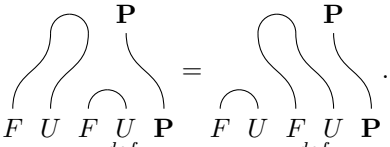
and



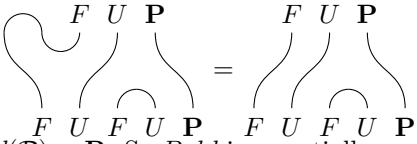
So **GL** is a category and setting $Bold(\mathcal{L}) = \mathbf{L}$, we have a functor $Bold : \mathbf{GL} \to$ C-**Prop**.

Since C-**Prop** is equivalent to the Eilenberg-Moore category of T then each prop $\mathbf{P}$ is a coequalizer of free props [3]. An explicit coequalizer is:

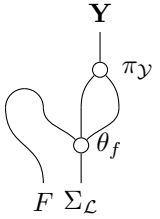
$$FUFU\mathbf{P} \underset{\epsilon_{FU\mathbf{P}}}{\overset{FU\epsilon_{\mathbf{P}}}{\rightrightarrows}} FU\mathbf{P} \xrightarrow{\epsilon_{\mathbf{P}}} \mathbf{P}$$

In fact graphically: 

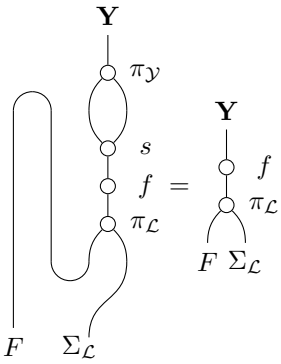
.

Let's take $\Sigma_{\mathcal{P}} \overset{def}{=} U\mathbf{P}$, $E_{\mathcal{P}} \overset{def}{=} UFU\mathbf{P}$, $r_{\mathcal{P}} \overset{def}{=} id_{FUFU\mathbf{P}}$ and $l_{\mathcal{P}} \overset{def}{=} F\eta_{U\epsilon_{\mathbf{P}}}$. We have: $\epsilon_{FU\mathbf{P}} \circ l_{\mathcal{P}} = \epsilon_{FU\mathbf{P}} \circ F\eta_{U\epsilon_{\mathbf{P}}} =$ 

$= FU\epsilon_{\mathbf{P}}$. Thus we can define $\mathcal{P} \overset{def}{=} (\Sigma_{\mathcal{P}}, E_{\mathcal{P}}, l_{\mathcal{P}}, r_{\mathcal{P}})$ and we have $Bold(\mathcal{P}) \simeq \mathbf{P}$. So $Bold$ is essentially surjective.

Let s be a section of $\pi_{\mathcal{Y}}$. Given a prop morphism $f : Bold(\mathcal{L}) \to Bold(\mathcal{Y})$ we define a translation $\theta_f : \mathcal{L} \to \mathcal{Y}$ by $\theta_f \overset{def}{=} Usf\pi_{\mathcal{L}} \circ \eta_{\Sigma_{\mathcal{L}}}$. We check the soundness condition: $\pi_{\mathcal{Y}} \circ (\epsilon_{F\Sigma_{\mathcal{Y}}} \circ F\theta_f) =$ 

$=$



$= f \circ \pi_{\mathcal{L}}$. This gives us $Bold(\theta_f) = f$. So $Bold$ is full. □

Proof of Lemma 2. Since coproducts commute with coequalizers we have the following coequalizer:

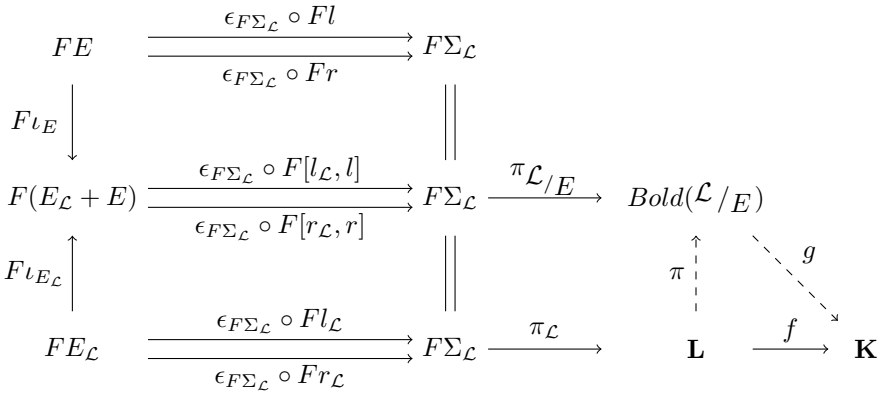
$$F(E_{\mathcal{L}} + E_{\mathcal{Y}}) \underset{(\epsilon_{F\Sigma_{\mathcal{L}}} \circ Fr_{\mathcal{L}}) + (\epsilon_{F\Sigma_{\mathcal{L}}} \circ Fr_{\mathcal{Y}})}{\overset{(\epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}}) + (\epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{Y}})}{\rightrightarrows}} F(\Sigma_{\mathcal{L}} + \Sigma_{\mathcal{Y}}) \xrightarrow{\pi_{\mathcal{L}} + \pi_{\mathcal{Y}}} \mathbf{L}{+}\mathbf{Y}$$

Furthermore: $F\iota_{\Sigma_{\mathcal{L}}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}} = \epsilon_{F\Sigma_{\mathcal{L}+\mathcal{Y}}} \circ FUF\iota_{\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}}$.

And so: $\epsilon_{F\Sigma_{\mathcal{L}+\mathcal{Y}}} \circ Fl_{\mathcal{L}+\mathcal{Y}} = \epsilon_{F\Sigma_{\mathcal{L}+\mathcal{Y}}} \circ F[UF(\iota_{\Sigma_{\mathcal{L}}}) \circ l_{\mathcal{L}}, UF(\iota_{\Sigma_{\mathcal{L}}}) \circ l_{\mathcal{Y}}] = [\epsilon_{F\Sigma_{\mathcal{L}+\mathcal{Y}}} \circ FUF(\iota_{\Sigma_{\mathcal{L}}}) \circ l_{\mathcal{L}}, \epsilon_{F\Sigma_{\mathcal{L}+\mathcal{Y}}} \circ FUF(\iota_{\Sigma_{\mathcal{L}}}) \circ l_{\mathcal{Y}}] = [F\iota_{\Sigma_{\mathcal{L}}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}}, F\iota_{\Sigma_{\mathcal{Y}}} \circ \epsilon_{F\Sigma_{\mathcal{Y}}} \circ Fl_{\mathcal{Y}}] = (\epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}}) + (\epsilon_{F\Sigma_{\mathcal{Y}}} \circ Fl_{\mathcal{Y}})$.

□

Proof of Lemma 3. We use the following diagram:



First we have $\pi_{\mathcal{L}/E} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}} = \pi_{\mathcal{L}/E} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fr_{\mathcal{L}}$ and thus there is a unique $\pi : \mathbf{L} \to Bold(\mathcal{L}/E)$ such that $\pi \circ \pi_{\mathcal{L}} = \pi_{\mathcal{L}/E}$. Then: $\pi \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl = \pi \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fr$.

It remains to show the universal property. Let $f : \mathbf{L} \to \mathbf{K}$ be a prop morphism such that: $f \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl = f \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fr$.

We also have $f \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fl_{\mathcal{L}} = f \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ Fr_{\mathcal{L}}$ and then the universal property of the coproduct gives us $f \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ F[l_{\mathcal{L}}, l] = f \circ \pi_{\mathcal{L}} \circ \epsilon_{F\Sigma_{\mathcal{L}}} \circ F[r_{\mathcal{L}}, r]$.

So there is a unique $g : Bold(\mathcal{L}/E) \to \mathbf{K}$ such that: $f \circ \pi_{\mathcal{L}} = g \circ \pi_{\mathcal{L}/E} = g \circ \pi \circ \pi_{\mathcal{L}}$. And since $\pi_{\mathcal{L}}$ is an epimorphism: $f = g \circ \pi$. □

A.2 Proofs for section 2

Proof of Lemma 2. We will show that for any diagram $\omega : a \to b$ we have $\mathbb{W} \vdash \omega = \Gamma_b \circ \sigma \circ \Delta_a$ where σ is a permutation. To do so we define, for each wire in the diagram, its **situation** as a couple of elements of $\{i, o, d, g\}$. The situation of a wire describes what the wire is linked to what: i stands for input, o for output, d for non identity divider, and g for non identity gatherer. For example, a wire which links an input

to an output has situation (i, o) and a wire linking a gatherer to a divider has situation (g, d). The possible situations for a simple wire are: (i, o), (i, g), (d, o), and (d, g). The possible situations for a big wire are the same plus (i, d), (d, d), (g, o), (g, g) and (g, d).

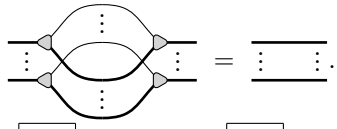
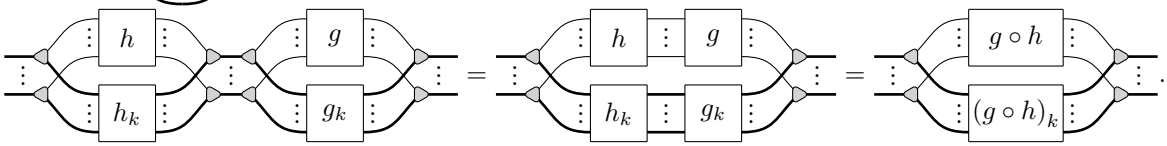
We say that a diagram is **expanded** if it contains no big wire in one of the **bad** situations which are (g, d) and (d, g).

The expanded conditions enforce a unique structure. In fact the only expanded diagrams are exactly the $\Gamma_b \circ \sigma \circ \Delta_a$. Thus it only remains to show that any diagram can be rewritten into an expanded one.

We proceed by induction on the size of the biggest big wire in a bad situation.

If there are no such wire then we are already in expanded form. Else, we consider the biggest wires in a bad situations. If a wire is in situation (g, d) then the elimination rule 2.1 can be applied and decreases strictly the size of the big wires in bad situations. If a wire is in situation (g, d) then the expantion rule 2.1 can be applied and decreases strictly the size of the big wires in bad situations. Thus all the wire in bad situations can be removed and replace by wire with strictly smaller size. Then by induction all diagrams can be rewritten into expanded form.

□

Proof of Lemma 4. By induction. S_1 is clearly a functor. The expansion equation ensures that $S_k(id_n) = id_{1\cdot n}$:  The elimination equation ensures that $S_k(g \circ h) = S_k(g) \circ S_k(h)$:  □

Proof of Lemma 5. Given a generator $x \in \Sigma_{\mathcal{P}}$ we have $S_1(\pi_{\mathcal{P}}(x)) = \pi_{\mathcal{SP}}(x) = \pi_{\mathcal{P}}$. □

Proof of Lemma 3. We use the same method as in the prof of the rewiring theorem. In $\mathcal{SL}$, there are new situations: s represent a simple generator and S. The new possible situations for a simple wires are (i, s), (d, s), (s, s), (s, g) and (s, o). For a big wire the new situations are (i, S), (d, S), (g, S), (S, S), (S, g), (S, d) and (S, o).

A diagram of $\mathcal{SL}$ is in expanded form if it contains no big wire in one of the bad situations which are (g, d), (d, g), (i, S), (d, S), (g, S), (S, S), (S, g), (S, d) and (S, o).

So an expanded diagram contains no big generators and is of the form $\omega = \Gamma_b \circ \nu \circ \Delta_a$. Where ν contains no big wire, so $S_1(\{\nu\}) = \nu$. Moreover $S_1(\{\omega\}) = S_1(\{\Gamma_b \circ \nu \circ \Delta_a\}) = S_1(\{\Gamma_b\}) \circ S_1(\{\nu\}) \circ S_1(\{\Delta_a\}) = S_1(\{\nu\}) = \nu$. So for an expanded diagram $\omega = \Gamma_b \circ S_1(\{\omega\}) \circ \Delta_a$.

It remains to show that any diagram can be rewritten in expanded form. We proceed by induction on the size of the biggest big wire in a bad situation.

If there is no such wire then we are already in expanded form. Else, we consider the biggest wires in a bad situations.

First we apply the expansion rule to remove all the biggest wires in situation (i, S), (d, S), (S, S), (S, g) and (S, o). If a wire is in situation (g, d) then the elimination rule 2.1 can be applied and decreases strictly the size of the big wires in bad situations. If a wire is in situation (g, d) then the expantion rule 2.1 can be applied and decreases strictly the size of the big wires in bad situations. If a wire is in situation (g, S) or (S, d) we apply the corresponding unfold equation. This decreases strictly the size of the big wires in bad situations. Thus all the wire in bad situations can be removed and replace by wire with strictly smaller size. Then by induction all diagrams can be rewritten into expanded form.

□

Proof of Lemma 6. The diagram clearly commutes. Now let $f : \mathbf{K} \to \mathbf{1}_{\mathbb{N}_0}$ and $g : \mathbf{K} \to \mathbf{L}$ be two functors such that $[\_] \circ f = ! \circ g$. We define a functor $h : \mathbf{K} \to \mathcal{SL}$. Given a morphism $t \in \mathbf{K}[a, b]$ we take $h(t) = \Gamma_{f(a)} \circ S_1(g(t)) \circ \Delta_{f(b)}$. This is well defined since $[f(a)] = g(a)$. We have $\{h(t)\} = \{\Gamma_{f(a)} \circ S_1(g(t)) \circ \Delta_{f(b)}\} =$

$\{S_1(g(t))\} = g(t)$ and $!(h(t)) = !(\Gamma_{f(a)} \circ S_1(g(t)) \circ \Delta_{f(b)}) = !_{f(a),f(b)} = f(t)$. Now let $l : \mathbf{K} \to \mathcal{S}\mathbf{L}$ be another functor such that $\{\_\} \circ l = g$ and $! \circ l = f$. we have $l(t) \in \mathcal{S}\mathbf{L}[l(a), l(b)]$ the structure theorem gives us: $l(t) = \Gamma_{l(a)} \circ S_1(\{l(t)\}) \circ \Delta_{l(b)} = \Gamma_{f(a)} \circ S_1(g(t)) \circ \Delta_{f(b)}$. So $l = h$. The diagram is a pullback square. □

Proof of Lemma 7. By construction $\mathbf{C}_p$ is the pullback of $! : \mathbf{C} \to \mathbf{N}$ and $[\_] : \mathbf{1}_{\mathbb{N}_0} \to \mathbf{N}$. Thus $[\![\_]\!]$ lift to a unique functor $[\![\_]\!]_p : \mathcal{L}\mathbf{L} \to \mathbf{C}_p$ satisfying $\{\_\} \circ [\![\_]\!]_p = [\![\_]\!] \circ \{\_\}$. This functor is an $\mathbb{N}_0$-colored prop morphism. Furthermore since $!$ and $\{\_\}$ are jointly monic then $[\![\_]\!]_p$ is faithful iff $[\![\_]\!]$ is faithful. □

Proof of Lemma 8. We consider the wire striper functor $\{\_\} : \mathcal{S}\mathbf{L} \to \mathbf{L}$. It is clearly essentially surjective. It is also full and faithful since it induces a bijection between $\mathcal{S}\mathbf{L}[\mathbf{a}, \mathbf{b}]$ and $\mathbf{L}[[\mathbf{a}], [\mathbf{b}]]$ by the structure theorem. So it is an equivalence of category and $\mathbf{L} \simeq \mathcal{S}\mathbf{L}$. □

A.3 Proofs for section 3

Proof of Lemma 4. For a diagram $\omega : \mathbf{a} \to \mathbf{b}$ we define $\omega' \stackrel{def}{=} \Delta_{\mathbf{a}} \circ \omega \circ \Gamma_{\mathbf{b}}$. We have $\Gamma_{\mathbf{a}} \circ B(O(\omega)) \circ \Delta_{\mathbf{b}} = \Gamma_{\mathbf{a}} \circ B(O(\Gamma_{\mathbf{a}} \circ \omega' \circ \Delta_{\mathbf{b}})) \circ \Delta_{\mathbf{b}} = \Gamma_{\mathbf{a}} \circ B(O(\omega')) \circ \Delta_{\mathbf{b}}$. Thus we only need to show that $B(O(\omega')) = \omega'$ for all $\omega' : n \cdot 1 \to m \cdot 1$.

A diagram of $Box\mathbf{P}$ of type $n \cdot 1 \to m \cdot 1$ is in **boxed form** if it is of the form $\Delta_m \circ \Box_f \circ \Gamma_n$.

A diagram in boxed form satisfies $B(O(\omega')) = \omega'$ thus we just have to show that any $\omega' : n \cdot 1 \to m \cdot 1$ can be rewrite into boxed form.

First we use the *swap* rule to transform all swaps into boxes. Then the diagram can be put in a sequence of the form $w_0 \circ b_0 \circ \cdots \circ w_h \circ b_h \circ w_{h+1}$, where the w_is are in $\mathbf{W}$ and the b_is are of the form $id_{\mathbf{c}} \otimes \Box_f \otimes id_{\mathbf{d}}$. We use the tensor rule on each b_i until we obtain a new sequence $w'_0 \circ \Box_{f_0} \circ \cdots \circ w'_h \circ \Box_{f_h} \circ w'_{h+1}$. The rewiring theorem gives us $\omega' = \Delta_m \circ \Box_{f_0} \circ \cdots \circ \Box_{f_h} \circ \Gamma_n$. Finally setting $f = f_0 \circ \cdots \circ f_h$ the composition rule gives $\omega' = \Delta_m \circ \Box_f \circ \Gamma_n$ a diagram in boxed form. □

Proof of Lemma 9. *Unbox* is an $\mathbb{N}_0$-colored prop morphism so it is essentially surjective.

Let $\alpha, \beta \in Box\mathbf{L}[\mathbf{a}, \mathbf{b}]$ be two diagrams such that $Unbox(\alpha) = Unbox(\beta)$. We have $S_1(O(\alpha)) = S_1(O(\beta))$, then applying the wire stripper functor $O(\alpha) = O(\beta)$. The structure theorem of $Box\mathbf{L}$ finally gives us $\alpha = \beta$. *Unbox* is faithful.

Given $\omega \in \mathcal{S}\mathbf{L}[\mathbf{a}, \mathbf{b}]$, the structure theorem gives us $\omega = \Gamma_{\mathbf{b}} \circ S_1(\{\omega\}) \circ \Delta_{\mathbf{a}}$

Then $Unbox(\Gamma_{\mathbf{b}} \circ B(\{\omega\}) \circ \Delta_{\mathbf{a}}) = \Gamma_{\mathbf{b}} \circ S_1(O(B(\{\omega\}))) \circ \Delta_{\mathbf{a}} = \Gamma_{\mathbf{b}} \circ S_1(\{\omega\}) \circ \Delta_{\mathbf{a}} = \omega$. *Unbox* is full. □